Title	**Non-clairvoyant scheduling for weighted flow time and energy on speed bounded processors**
Author(s)	**Chan, SH; Lam, TW; Lee, LK; Ting, HF; Zhang, P**
Citation	**CATS 2010, Computing: The Australasian Theory Symposium, Brisbane, Australia, 18-21 January 2010. In Chicago Journal of Theoretical Computer Science, 2011, p. article no. 1**
Issued Date	**2011**
URL	**http://hdl.handle.net/10722/140791**
Rights	**Creative Commons: Attribution 3.0 Hong Kong License**

CHICAGO JOURNAL OF THEORETICAL COMPUTER SCIENCE 2011, Article 1, pages 1–14
http://cjtcs.cs.uchicago.edu/

SPECIAL ISSUE FOR CATS 2010

Non-clairvoyant Scheduling for Weighted Flow Time and Energy on Speed Bounded Processors*

Sze-Hang Chan Tak-Wah Lam† Lap-Kei Lee Hing-Fung Ting Pan Zhang‡

Received: March 12, 2010; published: May 5, 2011.

Abstract: We consider the online scheduling problem of minimizing total weighted flow time plus energy in the dynamic speed scaling model, where a processor can scale its speed dynamically between 0 and some maximum speed T. In the past few years this problem has been studied extensively under the clairvoyant setting, which requires the size of a job to be known at release time [1, 4, 5, 8, 15, 18–20]. For the non-clairvoyant setting, despite its practical importance, the progress is relatively limited. Only recently an online algorithm LAPS is known to be $O(1)$-competitive for minimizing (unweighted) flow time plus energy in the infinite speed model (i.e., $T = \infty$) [11, 12]. This paper makes two contributions to the non-clairvoyant scheduling. First, we resolve the open problem that the unweighted result of LAPS can be extended to the more realistic model with bounded maximum speed. Second, we show that another non-clairvoyant algorithm WRR is $O(1)$-competitive when weighted flow time is concerned. Note that WRR is not as efficient as LAPS for scheduling unweighted jobs as WRR has a much bigger constant hidden in its competitive ratio.

*This is a corrected version of a paper with the same title in CATS 2010 [14]; in particular, Lemmas 2 and 4 of Section 3 and the ordering of jobs in the potential analysis of Section 4 were given incorrectly before and are fixed in this version. On the other hand, the conjecture, given in Section 5, about the generalization of LAPS to the weighted setting has recently been resolved [13].

†Research is partially supported by HKU Grant 21476018.

‡Research is partially supported by NSFC Grant 60970003.

DOI: 10.4086/cjtcs.2011.001

1 Introduction

Energy consumption has become an important concern for the design of modern microprocessors. Manufacturers like Intel and IBM are now producing processors that can support *dynamic speed scaling*, which would allow operating systems to manage the power by scaling the processor speed dynamically. Running jobs slower saves more energy, yet it takes longer time. Taking speed scaling and energy usage into consideration makes job scheduling more complicated than before. The challenge arises from the conflicting objectives of optimizing some quality of service (QoS) of the schedule and minimizing the energy usage.

The theoretical study of speed scaling was initiated by Yao, Demers and Shenker [24]. They considered a model where a processor can vary its speed s between 0 and infinity dynamically, and it consumes energy at the rate s^α, where α is a constant (commonly believed to be 2 or 3 [9, 22] for CMOS-based processors). Under this infinite speed model, Yao et al. studied the deadline scheduling and gave an online algorithm that is O(1)-competitive for minimizing the energy for completing all jobs. Algorithms with better ratios were later obtained by Bansal, Kimbrel and Pruhs [7] and Bansal, Chan, Pruhs and Katz [6]. The best ratio now is $4^\alpha/2\sqrt{e\alpha}$ for general α and is about 6.7 when $\alpha = 3$. The infinite speed model is a convenient model to work with. Among others, it allows an online algorithm to catch up arbitrarily fast and recover from any over-conservative decision on speed. However, this is not a practical model. Recently, Chan et al. [10] and Bansal et al. [4] have obtained several interesting results on speed scaling for a speed bounded processor, where the maximum speed T is a fixed constant.

Flow and energy. The study of speed scaling and energy-efficient scheduling goes beyond deadline scheduling. When scheduling jobs without deadlines, a commonly used QoS measure is the total flow time of jobs. The flow time (or simply the flow) of a job is the time elapsed since the job is released until it is completed. Note that job preemption is allowed, and a preempted job can be resumed later at the point of preemption. Assuming jobs are equally important, it is natural to find a schedule that minimizes the total flow time (which is also referred to as minimizing the total/average response time in the literature). When jobs have varying importance or weights, it is more meaningful to minimize the total weighted flow time.

Minimizing flow time and minimizing energy usage are orthogonal objectives. To understand their tradeoff, Albers and Fujiwara [1] initiated the study of minimizing a linear combination of flow and energy. The intuition is that, from an economic viewpoint, users are willing to pay a certain (say, ρ) units of energy to reduce one unit of flow time. By changing the units of time and energy, one can further assume that $\rho = 1$ and thus would like to minimize flow time plus energy, or in general, weighted flow time plus energy.

Clairvoyant scheduling for flow plus energy. The problem of minimizing flow plus energy has attracted a lot of attention [1,4,5,8,12,18–20]. These works mainly focus on the clairvoyant setting which assumes that the size of a job is known when the job is released. The work of Albers and Fujiwara [1] focused on jobs of unit size. Bansal, Pruhs and Stein [8] were the first to consider jobs of arbitrary sizes. In the infinite speed model, they gave an algorithm that is $O((\frac{\alpha}{\ln \alpha})^2)$-competitive for minimizing weighted

Key words and phrases: online algorithms, non-clairvoyant scheduling, speed scaling, energy, flow time

flow plus energy. Bansal, Chan, Lam and Lee [4] later adapted the BPS algorithm to the bounded speed model. The competitive ratio remains $O((\frac{\alpha}{\ln \alpha})^2)$ when the algorithm uses a processor with maximum speed $(1+\frac{\ln \alpha - \ln\ln \alpha}{\alpha})T$.[1] Very recently, Bansal, Chan and Pruhs [5] improved the analysis of BPS; their work implies that BPS is indeed $O(\frac{\alpha}{\ln \alpha})$-competitive, when the maximum speed is relaxed as before. It is worth-mentioning that the recent lower bound result on weighted flow [3] implies that without relaxing the maximum speed, no online algorithm can be constant competitive (in terms of α) for weighted flow plus energy. However, the extra speed requirement does not apply to unweighted flow.

A drawback of the BPS algorithm is that it scales the speed according to the fraction of unfinished work and thus keeps changing the speed continuously over time. It is practically more desirable to have an algorithm that changes the speed at discrete times (say, at job arrival or completion) . Recently, focusing on (unweighted) flow plus energy, Lam et al. [18] studied another speed scaling algorithm AJC (Active Job Count), which scales the speed as a function of the number of active jobs (i.e. unfinished jobs). In other words, AJC changes the speed only when a job arrives or finishes. AJC when coupled with the job selection algorithm SRPT (shortest remaining processing time) is indeed $O(\frac{\alpha}{\log \alpha})$-competitive for (unweighted) flow plus energy. This result holds in both the infinite and bounded speed models. For the latter, unlike BPS, AJC does not demand relaxation of maximum speed. Recently, Bansal, Chan and Pruhs [5] adapted AJC and gave a tighter analysis; they showed that the competitive ratio is 3 for minimizing (unweighted) flow plus energy (even when α is as small as 3, this result is still better than the $O(\frac{\alpha}{\log \alpha})$ bound in [18], which is equal to 3.25). Again, no extra maximum speed is needed. More recently, the analysis is further tightened by [2] and the competitive ratio is reduced to 2.

For weighted flow plus energy, it has remained an open problem whether AJC (or any speed scaling algorithm that changes the speed at discrete times) can lead to a competitive guarantee.

Non-clairvoyant scheduling for flow plus energy. All of the above results assume clairvoyance. In the non-clairvoyant setting, the size of a job is not known when the job is released; it is only known when the job is completed. This is a natural assumption from the viewpoint of operating systems. Non-clairvoyant flow time scheduling (on a fixed-speed processor) has been an interesting problem itself (e.g., [17, 21]). Chan et al. [12] initiated the study of non-clairvoyant speed scaling. Under the infinite speed model, they consider an algorithm LAPS (Latest Arrival Processor Sharing) which scales the speed as AJC and selects some most recently released jobs to share the processor. LAPS is shown to be $4\alpha^3(1+(1+3/\alpha)^\alpha) = O(\alpha^3)$-competitive for (unweighted) flow plus energy. Furthermore, they showed that no algorithm can be $O(\alpha^{1/3-\varepsilon})$-competitive for any $\varepsilon > 0$, illustrating that the non-clairvoyant setting is more difficult than the clairvoyant setting. Recently, Chan et al. [11] improved the analysis of LAPS and reduce the competitive ratio to $O(\frac{\alpha^2}{\log \alpha})$. Yet, not much is known for the bounded speed model, let alone weighted flow plus energy.

Our contributions. This paper considers non-clairvoyant scheduling on a processor whose maximum speed T is a fixed constant. In the first part, we adapt the algorithm LAPS to run on such a processor and show that it is $8\alpha^2 = O(\alpha^2)$-competitive for (unweighted) flow plus energy when the maximum speed is relaxed to $(1+\frac{1}{\alpha-1})T$. Note that unlike the clairvoyant setting, even for unweighted jobs, extra maximum speed is necessary to achieve constant competitiveness. This inherits from the lower bound result on non-clairvoyant (fixed-speed) scheduling by Motwani, Philips and Torng [21].

[1] In general, when using a processor with maximum speed $(1+\varepsilon)T$ for any $\varepsilon > 0$, the competitive ratio is $\max\{(1+1/\varepsilon), (1+\varepsilon)^\alpha\}(2+o(1))\alpha/\ln \alpha$.

In the clairvoyant setting, existing results on the bounded speed model take advantage of a local property that the online algorithm in concern accumulates at most $O(T^\alpha)$ jobs more than the optimal offline algorithm [4,5,18]. With this local property, it is relatively easy to adapt the analysis in the infinite speed model. In the non-clairvoyant setting, such a local property is no longer valid for algorithms like LAPS. To analyze these algorithms in the bounded speed model, we exploit a more "global" accounting of the rate of change of flow plus energy. Instead of using the above property, we integrate the maximum speed constraint into the potential analysis.

The second result of this paper concerns the more difficult general case where jobs have arbitrary weights. Under the bounded speed model, we give the first competitive algorithm called WRR (Weighted Round Robin) for weighted flow plus energy; the competitive ratio is $O(3^\alpha/\varepsilon)$ when using a processor with maximum speed $(3+\varepsilon)T$, where $0 < \varepsilon \leq \frac{3}{\alpha}$. Motivated by AJC, WRR uses a generalized AJC for speed scaling; i.e., the speed is a function of the total weight (instead of the number) of active jobs. Recall that all existing clairvoyant results [4,8] on weighted flow plus energy are based on the BPS algorithm, which scales the speed continuously. Our result of WRR gives, as a by-product, the first competitive clairvoyant algorithm for weighted flow plus energy that changes the speed discretely, but the competitive ratio is way worse than that of BPS.

2 Definitions and Notations

We study job scheduling on a single processor. Jobs arrive over time in an online fashion; we have no information about a job before it arrives. For any job j, we use $r(j)$ and $p(j)$ to denote its release time and work requirement (or size). In some case, each job j may have a weight $w(j)$. We consider the *non-clairvoyant* setting, in which when a job j arrives, we only know its weight $w(j)$ (if any) but not its size $p(j)$. And $p(j)$ is known only when j is completed. At any time, the processor can scale its speed between 0 and a *maximum speed* T. When running at speed s, the processor processes s units of work per unit time and consumes energy at the rate s^α, where $\alpha > 1$ is a fixed constant. Preemption is allowed; a job can be preempted and later resumed at the point of preemption without any penalty.

Flow and energy. Consider any job set I and some schedule S of I. At any time t, for any job j, we let $q(j,t)$ be the remaining work of j at t. A job j is an *active job* if it has been released but not yet completed, i.e., $r(j) \leq t$ and $q(j,t) > 0$. The flow $F(j)$ of a job j is the time elapsed since j arrives and until it is completed. The *total flow* F is equal to $\sum_{j\in I} F(j)$, which is equivalent to $\int_0^\infty n(t)\,\mathrm{d}t$, where $n(t)$ is the total number of active jobs at time t. The energy usage E is $\int_0^\infty (s(t))^\alpha\,\mathrm{d}t$, where $s(t)$ is the processor speed at time t. The objective is to minimize the sum of total flow and energy usage, denoted by $G = F + E$.

In general, when jobs have different weights, we generalize the notion of total flow as follows. The *total weighted flow* $\widehat{F}$ is equal to $\sum_{j\in I} w(j)F(j)$, or equivalently, $\int_0^\infty w(t)\,\mathrm{d}t$, where $w(t)$ is the total weight of active jobs at time t. The objective becomes minimizing total weighted flow plus energy, denoted by $\widehat{G} = \widehat{F} + E$.

3 Minimizing unweighted flow plus energy with bounded maximum speed

In this section, we consider jobs without weights and aim at minimizing total flow plus energy in the bounded speed model. As mentioned in the introduction, no online algorithm can achieve constant competitiveness when its maximum speed is the same as the optimal offline algorithm OPT (which is denoted T below); thus, we consider allowing the online algorithm to use slightly higher maximum speed. We adapt the non-clairvoyant algorithm LAPS (Latest Arrival Processor Sharing) which was first given in [12] for the infinite speed model. When using a processor with maximum speed $(1+\delta)T$, where $\delta = \frac{1}{\alpha-1}$, this algorithm is $O(\alpha^2)$-competitive for flow plus energy.

Below is the definition of LAPS, which assumes using a processor with maximum speed $(1+\delta)T$ for some $\delta > 0$.

> **Algorithm LAPS.** Let $0 < \beta \leq 1$ be any real. Consider any time t. The processor speed is set to $s_a(t) = \min((n_a(t))^{1/\alpha}, (1+\delta)T)$, where $n_a(t)$ is the total number of active jobs at t. The processor processes the $\lceil \beta n_a(t) \rceil$ active jobs with the latest release time (ties are broken by job ids) by splitting the processor speed equally among these jobs.

We compare LAPS with an optimal offline algorithm OPT using a processor with maximum speed T. Our main result is the following theorem.

Theorem 3.1. *When $\delta = \frac{1}{\alpha-1}$ and $\beta = \frac{1}{2\alpha}$,* LAPS *is $8\alpha^2$-competitive for (unweighted) flow plus energy, using a processor with maximum speed $(1+\delta)T$.*

To prove Theorem 3.1, our analysis exploits amortization and potential functions (e.g., [8, 10]). Let $G_a(t)$ and $G_o(t)$ denote the flow plus energy incurred up to time t by LAPS and OPT, respectively. We drop the parameter t when it is clear that t is the current time. To show that LAPS is c-competitive for some constant $c \geq 1$, it suffices to define a potential function $\Phi(t)$ such that the following conditions hold: (i) $\Phi = 0$ before any job is released and after all jobs are completed; (ii) Φ is a continuous function except at some discrete times (e.g., when a job arrives, or when a job is completed by LAPS or OPT), and Φ does not increase at such times; (iii) at any other time, $\frac{dG_a(t)}{dt} + \gamma \frac{d\Phi(t)}{dt} \leq c \cdot \frac{dG_o(t)}{dt}$, where γ is a positive constant (to be set to 4α). Condition (iii) is also known as the running condition. The sufficiency of these conditions for proving c-competitiveness follows from integrating them over time.

Potential function $\Phi(t)$. Consider any time t. Let $n_a(t)$ and $n_o(t)$ be the number of active jobs in LAPS and OPT, respectively. Let $j_1, j_2, \ldots, j_{n_a(t)}$ be all the active jobs in LAPS, which are ordered by release times such that $r(j_1) \leq r(j_2) \leq \cdots \leq r(j_{n_a(t)})$ (ties are broken by job ids). For any job j, let $q_a(j,t)$ and $q_o(j,t)$ be the remaining work of job j in LAPS and OPT, respectively. For each j_i, let $x_i = \max\{q_a(j_i,t) - q_o(j_i,t), 0\}$ which is the amount of work of j_i in LAPS that is lagging behind OPT. We call a job j_i *lagging* if $x_i > 0$. The potential function $\Phi(t)$ is defined as follows.

$$\Phi(t) = \sum_{i=1}^{n_a(t)} c_i \cdot x_i$$

$$\text{where } c_i = \begin{cases} i^{1-1/\alpha} & \text{if } i^{1/\alpha} \leq (1+\delta)T; \\ i/((1+\delta)T) & \text{otherwise.} \end{cases}$$

We call c_i the *coefficient* of j_i. Note that c_i is monotonically increasing.

We first check Conditions (i) and (ii). Condition (i) holds since $\Phi = 0$ before any job is released and after all jobs are completed. Now we check Condition (ii). When a job j arrives, j must be non-lagging and the coefficients of all existing jobs of LAPS remain the same, so Φ does not change. When OPT completes a job, Φ does not change. When LAPS completes a job, the coefficient of any other job either stays the same or decreases, so Φ does not increase.

It remains to check the running condition (Condition (iii)). Consider any time t when Φ does not have discrete change. Let s_a and s_o be the current speeds of LAPS and OPT, respectively. Then $\frac{dG_a}{dt} = n_a + s_a^\alpha$ and $\frac{dG_o}{dt} = n_o + s_o^\alpha$. Let ℓ be the number of lagging jobs that LAPS is processing. Note that $\ell \le \lceil \beta n_a \rceil$. For convenience, we further define another real number $\phi \le \beta$ such that $(\beta - \phi)n_a$ is an integer equal to $\lceil \beta n_a \rceil - \ell$. Note that ϕ can be less than zero if $\ell = 0$.

To bound the rate of change of Φ, we consider how Φ changes in an infinitesimal amount of time (from t to $t + dt$), first due to LAPS only (Lemma 3.2), and then due to OPT (Lemma 3.3). We denote the rate of change of Φ due to LAPS and OPT by $\frac{d\Phi_a}{dt}$ and $\frac{d\Phi_o}{dt}$, respectively. Note that $\frac{d\Phi}{dt} = \frac{d\Phi_a}{dt} + \frac{d\Phi_o}{dt}$.

Lemma 3.2. $\frac{d\Phi_a}{dt} \le -\frac{\phi}{\beta}(1-\beta)n_a$.

Proof. LAPS is processing $\lceil \beta n_a \rceil$ jobs and ℓ of them are lagging jobs. For each of these lagging jobs j_i, its lagging size x_i is changing at the rate of $-s_a/\lceil \beta n_a \rceil$ (we only consider the change due to LAPS). For a non-lagging job j_i, x_i does not change.

First, consider the case that $\ell \ge 1$. To upper bound $\frac{d\Phi_a}{dt}$, the worst case is that the ℓ lagging jobs have the smallest coefficients among the $\lceil \beta n_a \rceil$ latest released jobs, i.e., the jobs are $\{j_{n_a - \lceil \beta n_a \rceil + 1}, \cdots, j_{n_a - \lceil \beta n_a \rceil + \ell}\}$. On the other hand, as c_i is monotonically increasing, for any integers $a < b$, we have $\sum_{i=a}^{b} c_i \ge (b - a + 1)c_a$. Then

$$\begin{aligned}\frac{d\Phi_a}{dt} &\le -\sum_{i=n_a-\lceil \beta n_a \rceil+1}^{n_a-\lceil \beta n_a \rceil+\ell} c_i \cdot \frac{s_a}{\lceil \beta n_a \rceil} \\ &\le -\ell(c_{n_a-\lceil \beta n_a \rceil+1})\frac{s_a}{\lceil \beta n_a \rceil}.\end{aligned}$$

Recall that $s_a = \min(n_a^{1/\alpha}, (1+\delta)T)$. By the definition of c_i, we have $c_i s_a \ge i$ for any $1 \le i \le n_a$. Furthermore, $\ell = \lceil \beta n_a \rceil - (\beta - \phi)n_a$ and hence $n_a - \lceil \beta n_a \rceil + \ell = n_a - (\beta - \phi)n_a$. Thus,

$$\begin{aligned}\frac{d\Phi_a}{dt} &\le -\left(\frac{\ell}{\lceil \beta n_a \rceil}\right)(n_a - \lceil \beta n_a \rceil + 1) \\ &\le \left(\frac{-\lceil \beta n_a \rceil + (\beta-\phi)n_a}{\lceil \beta n_a \rceil}\right)(n_a - \beta n_a) \\ &\le (-1 + \tfrac{(\beta-\phi)n_a}{\beta n_a})(1-\beta)n_a \quad (\text{since } (\beta - \phi) \ge 0) \\ &= -\tfrac{\phi}{\beta}(1-\beta)n_a \ .\end{aligned}$$

It remains to consider the case that $\ell = 0$. In this case, $\frac{d\Phi_a}{dt} = 0$. Recall that $0 < \beta \le 1$. Note also that $\phi n_a = \beta n_a - \lceil \beta n_a \rceil + \ell \le 0$, i.e., $\phi \le 0$. Then $-\frac{\phi}{\beta}(1-\beta)n_a \ge 0 = \frac{d\Phi_a}{dt}$. □

Lemma 3.3. *Assume that* $\delta = \frac{1}{\alpha-1}$. *Then* $\frac{d\Phi_o}{dt} \le (1-\frac{1}{\alpha})n_a + \frac{1}{\alpha}s_o^\alpha$.

Proof. To upper bound $\frac{d\Phi_o}{dt}$, the worst case is that OPT is processing the job j_{n_a} with the largest coefficient c_{n_a}. Thus, $\frac{d\Phi_o}{dt} \le c_{n_a}s_o$.

If $n_a^{1/\alpha} \le (1+\delta)T$, we have $c_{n_a} = n_a^{1-1/\alpha}$ and hence $\frac{d\Phi_o}{dt} \le n_a^{1-1/\alpha}s_o$. We apply the Young's Inequality [23], which is stated in Lemma 3.5 below, by setting $p = 1/(1-\frac{1}{\alpha})$, $q=\alpha$, $x = n_a^{1-1/\alpha}$ and $y = s_o$. Then $\frac{d\Phi_o}{dt} \le n_a^{1-1/\alpha}s_o \le (1-\frac{1}{\alpha})n_a + \frac{1}{\alpha}s_o^\alpha$.

If $n_a^{1/\alpha} > (1+\delta)T$, we have $c_{n_a} = \frac{n_a}{(1+\delta)T}$. Recall that $s_o \le T$ and $\delta = \frac{1}{\alpha-1}$. We conclude that $\frac{d\Phi_o}{dt} \le \left(\frac{n_a}{(1+\delta)T}\right)s_o \le \left(\frac{n_a}{1+\delta}\right) = (1-\frac{1}{\alpha})n_a$. The lemma thus follows. □

We are now ready to prove the running condition, which together with Conditions (i) and (ii), implies Theorem 3.1.

Lemma 3.4. *Assume that* $\delta = \frac{1}{\alpha-1}, \beta = \frac{1}{2\alpha}$, *and* $\gamma = 4\alpha$. *At any time when* Φ *does not have discrete change,* $\frac{dG_a}{dt} + \gamma\frac{d\Phi}{dt} \le 8\alpha^2 \cdot \frac{dG_o}{dt}$.

Proof. We will show an equivalent version of the inequality, $\frac{dG_a}{dt} + \gamma\frac{d\Phi}{dt} - 8\alpha^2 \cdot \frac{dG_o}{dt} \le 0$.

LAPS is processing $\lceil \beta n_a \rceil - \ell$ non-lagging jobs, which are also active jobs in OPT. Thus, $n_o \ge \lceil \beta n_a \rceil - \ell = (\beta-\phi)n_a$. Note that $\frac{dG_a}{dt} = n_a + s_a^\alpha \le 2n_a$, and $\frac{dG_o}{dt} = n_o + s_o^\alpha \ge (\beta-\phi)n_a + s_o^\alpha$. Then

$$\frac{dG_a}{dt} + \gamma\frac{d\Phi}{dt} - 8\alpha^2 \cdot \frac{dG_o}{dt} \quad \le \quad 2n_a + \gamma\frac{d\Phi}{dt} - 8\alpha^2(\beta-\phi)n_a - 8\alpha^2 s_o^\alpha \ .$$

By Lemmas 3.2 and 3.3, $\gamma\frac{d\Phi}{dt} \le \gamma\cdot(1-\frac{1}{\alpha})n_a + \frac{\gamma}{\alpha}s_o^\alpha - \gamma\frac{\phi}{\beta}(1-\beta)n_a$. Then,

$$\frac{dG_a}{dt} + \gamma\frac{d\Phi}{dt} - 8\alpha^2 \cdot \frac{dG_o}{dt} \quad \le \quad 2n_a + \gamma\cdot(1-\frac{1}{\alpha})n_a + \frac{\gamma}{\alpha}s_o^\alpha - \gamma\frac{\phi}{\beta}(1-\beta)n_a - 8\alpha^2(\beta-\phi)n_a - 8\alpha^2 s_o^\alpha \ .$$

We set $\beta = \frac{1}{2\alpha}$ and $\gamma = 4\alpha$.

$$\begin{aligned}
\frac{dG_a}{dt} + \gamma\frac{d\Phi}{dt} - 8\alpha^2 \cdot \frac{dG_o}{dt} &\le (4-8\alpha^2)s_o^\alpha + n_a\left[2 + (4\alpha-4) - \phi(8\alpha^2-4\alpha) - (4\alpha - 8\alpha^2\phi)\right] \\
&\le n_a(-2+4\alpha\phi) \\
&\le 0 \qquad (\text{since } \phi \le \beta = \frac{1}{2\alpha}) \ .
\end{aligned}$$

□

Below is the formal statement of Young's Inequality, which is used in the proof of Lemma 3.3.

Lemma 3.5 (Young's Inequality [23])**.** *For positive reals* p,q,x,y *where* $\frac{1}{p}+\frac{1}{q} = 1$, $xy \le \frac{1}{p}x^p + \frac{1}{q}y^q$.

4 Minimizing weighted flow plus energy with bounded maximum speed

In this section, we consider jobs with arbitrary weights and give a non-clairvoyant algorithm WRR (Weighted Round Robin) that is $O(\alpha 3^\alpha)$-competitive for weighted flow plus energy, when using a processor with maximum speed $(3+\varepsilon)T$, where $\varepsilon = \frac{3}{\alpha}$. The algorithm WRR scales its speed based on the total weight of active jobs and shares the processor among the active jobs according to the ratio of their weights. Below is the definition of WRR, which assumes using a processor with maximum speed $(3+\varepsilon)T$ for any $\varepsilon > 0$.

> **Algorithm WRR.** Consider any time t. The processor speed is set to $s_a(t) = (3+\varepsilon) \cdot \min((w_a(t))^{1/\alpha}, T)$, where $w_a(t)$ is the total weight of active jobs at t. The processor processes all active jobs such that each active job j receives processor speed equal to $s(t) \cdot (w(j)/w_a(t))$.

We compare WRR against an optimal offline algorithm OPT that uses a processor with maximum speed T. Our main result is the following theorem.

Theorem 4.1. *Using a processor with maximum speed $(3+\varepsilon)T$, where $0 < \varepsilon \le \frac{3}{\alpha}$,* WRR *is c-competitive for weighted flow plus energy, where $c = (\frac{18}{\varepsilon}+4)(1+(3+\varepsilon)^\alpha) \le (\frac{18}{\varepsilon}+4)(1+3^\alpha e) = O(3^\alpha/\varepsilon)$.*

Notice that when $\varepsilon = \frac{3}{\alpha}$, the competitive ratio in the above theorem becomes $(6\alpha+4)(1+3^\alpha(1+\frac{1}{\alpha})^\alpha) = O(\alpha 3^\alpha)$. The rest of this section is devoted to proving Theorem 4.1. Let $\widehat{G}_a(t)$ and $\widehat{G}_o(t)$ be the weighted flow plus energy incurred up to time t by WRR and OPT, respectively. We drop the parameter t when it is clear that t is the current time. Similar to Section 3, to prove that WRR is c-competitive, we derive a potential function $\Phi(t)$ that satisfies the following conditions: (i) $\Phi = 0$ before any job is released and after all jobs are completed; (ii) Φ is a continuous function except at some discrete times where Φ does not increase; (iii) *Running condition*: at any other time, $\frac{d\widehat{G}_a(t)}{dt} + \gamma\frac{d\Phi(t)}{dt} \le c \cdot \frac{d\widehat{G}_o(t)}{dt}$, where γ is a positive constant (to be set to $(2-\frac{1}{\alpha})(1+(3+\varepsilon)^\alpha)$).

Potential function $\Phi(t)$. Consider any time t. For any job j, let $q_a(j,t)$ and $q_o(j,t)$ be the remaining work of j at t in WRR and OPT, respectively. An active job j in WRR is *lagging* if WRR has processed less on j than OPT at time t. Let $L = \{j_1, j_2, \ldots, j_\ell\}$ be the set of lagging jobs in WRR, ordered in ascending order of the latest time when the job becomes lagging. For each $j_i \in L$, let $x_i = q_a(j_i,t) - q_o(j_i,t)$; note that $x_i > 0$. We define the potential function $\Phi(t)$ as follows.

$$\Phi(t) = \sum_{i=1}^{\ell} c_i \cdot x_i \quad \text{where } c_i = \begin{cases} \left(\sum_{k=1}^{i} w(j_k)\right)^{1-1/\alpha} & \text{if } \sum_{k=1}^{i} w(j_k) \le T^\alpha; \\ \frac{2\alpha}{2\alpha-1}\left(\sum_{k=1}^{i} \frac{w(j_k)}{T}\right) & \text{otherwise.} \end{cases}$$

We call c_i the *coefficient* of j_i. Note that c_i is monotonically increasing with i.

We first check Conditions (i) and (ii). Condition (i) holds since $\Phi = 0$ before any job is released and after all jobs are completed. Now we show that Condition (ii) holds. When a job j_i joins L, x_i tends to

zero and j_i must be at the end of L, so the coefficients of all other jobs do not change and Φ does not change. When a job j_i leaves L (e.g., WRR completes j_i), x_i must be zero and the coefficient of any other lagging job either stays the same or decreases, so Φ does not increase.

It remains to check the running condition (Condition (iii)). Consider any time t when Φ does not have discrete change. Let w_a and w_o be the total weight of active jobs at t in WRR and OPT, respectively. Let $w_\ell = \sum_{i=1}^{\ell} w(j_i)$ be the total weight of jobs in L. Note that $w_\ell \le w_a$. Furthermore, let s_a and s_o be the current speeds of WRR and OPT, respectively. As stated in Section 2, $\frac{d\widehat{G}_a}{dt} = w_a + s_a^\alpha$ and $\frac{d\widehat{G}_o}{dt} = w_o + s_o^\alpha$. We will divide the analysis into cases depending on whether w_ℓ is small or big.

To bound the rate of change of Φ, we consider how Φ changes first due to OPT only (Lemma 4.2) and then due to WRR (Lemma 4.3). We denote the rate of change of Φ due to OPT and WRR by $\frac{d\Phi_o}{dt}$ and $\frac{d\Phi_a}{dt}$, respectively. Note that $\frac{d\Phi}{dt} = \frac{d\Phi_o}{dt} + \frac{d\Phi_a}{dt}$.

Lemma 4.2. *If $w_\ell \le T^\alpha$, $\frac{d\Phi_o}{dt} \le \frac{1}{\alpha} s_o^\alpha + (\frac{\alpha-1}{\alpha}) w_\ell$; if $w_\ell > T^\alpha$, $\frac{d\Phi_o}{dt} \le (\frac{2\alpha}{2\alpha-1}) w_\ell$.*

Proof. To upper bound $\frac{d\Phi_o}{dt}$, observe that the worst case is when OPT is processing the job j_ℓ with the largest coefficient c_ℓ. Then x_i is increasing at the rate of s_o (we only consider the change due to OPT) and hence $\frac{d\Phi_o}{dt} \le c_\ell s_o$. When $w_\ell \le T^\alpha$, $c_\ell = w_\ell^{1-1/\alpha}$ and thus $\frac{d\Phi_o}{dt} \le w_\ell^{1-1/\alpha} s_o$. We apply the Young's Inequality (Lemma 3.5 in Section 3) with $p = \alpha$, $q = \alpha/(\alpha-1)$, $x = s_o$ and $y = w_\ell^{1-1/\alpha}$. Then we have

$$\frac{d\Phi_o}{dt} \le \frac{1}{\alpha} s_o^\alpha + (\frac{\alpha-1}{\alpha}) w_\ell \ .$$

When $w_\ell > T^\alpha$, $c_\ell = (\frac{2\alpha}{2\alpha-1})(\frac{w_\ell}{T})$. Since $s_o \le T$, $\frac{d\Phi_o}{dt} \le c_\ell s_o \le c_\ell T \le (\frac{2\alpha}{2\alpha-1}) w_\ell$. □

Lemma 4.3. *If $w_\ell \le T^\alpha$, $\frac{d\Phi_a}{dt} \le -(\frac{\alpha}{2\alpha-1}) w_\ell^{2-1/\alpha} (\frac{s_a}{w_a})$; if $w_\ell > T^\alpha$, $\frac{d\Phi_a}{dt} \le -(\frac{\alpha}{2\alpha-1})(3+\varepsilon)(\frac{w_\ell^2}{w_a})$.*

Proof. To upper bound $\frac{d\Phi_a}{dt}$, note that each job $j_i \in L$ is being processed at the rate of $s_a \cdot \frac{w(j_i)}{w_a}$ (we only consider the change due to WRR), and thus x_i is changing at the rate of $-s_a \cdot \frac{w(j_i)}{w_a}$. To ease discussion, let $y_i = \sum_{k=1}^{i} w(j_k)$. Note that $y_0 = 0$, $y_\ell = w_\ell$, and for any $1 \le i \le \ell$, $y_i - y_{i-1} = w(j_i)$.

First, consider $w_\ell \le T^\alpha$. In this case, for each job $j_i \in L$, $c_i = y_i^{1-1/\alpha}$.

$$\begin{aligned}
\frac{d\Phi_a}{dt} &= \sum_{i=1}^{\ell} y_i^{1-1/\alpha} \cdot \left(-s_a \cdot \frac{w(j_i)}{w_a}\right) \\
&= -\frac{s_a}{w_a} \sum_{i=1}^{\ell} y_i^{1-1/\alpha} \cdot (y_i - y_{i-1}) \qquad \text{(since } y_i - y_{i-1} = w(j_i)\text{)} \\
&\le -\frac{s_a}{w_a} \sum_{i=1}^{\ell} \int_{y_{i-1}}^{y_i} x^{1-1/\alpha}\, dx \qquad \text{(since } x^{1-1/\alpha} \text{ is monotonically increasing)} \\
&\le -\frac{s_a}{w_a} \int_0^{y_\ell} x^{1-1/\alpha}\, dx \\
&= -\frac{s_a}{w_a} (\frac{y_\ell^{2-1/\alpha}}{2-1/\alpha}) \\
&= -(\frac{\alpha}{2\alpha-1}) w_\ell^{2-1/\alpha} (\frac{s_a}{w_a})
\end{aligned}$$

Next, consider $w_\ell > T^\alpha$. In this case, $w_{\rm a} \geq w_\ell > T^\alpha$ and hence $s_{\rm a} = (3+\varepsilon)\cdot\min(w_{\rm a}^{1/\alpha}, T) = (3+\varepsilon)T$. Note that $y_\ell = w_\ell > T^\alpha$. We let $g < \ell$ be the largest integer such that $y_g \leq T^\alpha$. Then

$$\begin{aligned}
\tfrac{{\rm d}\Phi_{\rm a}}{{\rm d}t} &= \sum_{i=1}^{\ell} c_i \cdot \left(-s_{\rm a}\cdot\frac{w(j_i)}{w_{\rm a}}\right) \\
&= -\left(\sum_{i=1}^{g} w(j_i) y_i^{1-1/\alpha} + \sum_{i=g+1}^{\ell} \frac{2\alpha}{2\alpha-1}\frac{w(j_i)y_i}{T}\right)\cdot\left(\frac{s_{\rm a}}{w_{\rm a}}\right) \\
&\leq -\left(\int_0^{y_g} x^{1-1/\alpha}\,{\rm d}x + \frac{2\alpha}{(2\alpha-1)T}\int_{y_g}^{y_\ell} x\,{\rm d}x\right)\cdot\left(\frac{s_{\rm a}}{w_{\rm a}}\right) \\
&= -\left((\frac{\alpha}{2\alpha-1})y_g^{2-1/\alpha} + \frac{\alpha}{(2\alpha-1)T}(y_\ell^2 - y_g^2)\right)\cdot\left(\frac{s_{\rm a}}{w_{\rm a}}\right) \\
&\leq -\left(\frac{\alpha}{(2\alpha-1)T}(y_g^2 + y_\ell^2 - y_g^2)\right)\cdot\left(\frac{(3+\varepsilon)T}{w_{\rm a}}\right) \quad \text{(since } y_g^{1/\alpha} \leq T \text{ and } s_{\rm a} = (3+\varepsilon)T\text{)} \\
&= -(\frac{\alpha}{2\alpha-1})(3+\varepsilon)(\frac{y_\ell^2}{w_{\rm a}}) \\
&= -(\frac{\alpha}{2\alpha-1})(3+\varepsilon)(\frac{w_\ell^2}{w_{\rm a}})\ . \qquad \square
\end{aligned}$$

We are ready to show the following running condition, which together with Conditions (i) and (ii), implies Theorem 4.1.

Lemma 4.4. *Assume that* $\gamma = (2-\frac{1}{\alpha})(1+(3+\varepsilon)^\alpha)$. *At any time when* Φ *does not have discrete change,* $\frac{{\rm d}\widehat{G}_{\rm a}}{{\rm d}t} + \gamma\frac{{\rm d}\Phi}{{\rm d}t} \leq c\cdot\frac{{\rm d}\widehat{G}_{\rm o}}{{\rm d}t}$, *where* $c = (\frac{18}{\varepsilon}+4)(1+(3+\varepsilon)^\alpha)$.

Proof. The analysis is divided into three cases depending on whether $w_{\rm a} > T^\alpha$ and whether $w_\ell > T^\alpha$. In each case, we further divide the analysis depending on whether $w_\ell > (1-\beta)w_{\rm a}$, where $\beta = \varepsilon/(6+2\varepsilon)$. It is useful to note that $(3+\varepsilon)(1-\beta)^2 \geq 3$.

Case 1: $w_{\rm a} \leq T^\alpha$**.** In this case, $s_{\rm a} = (3+\varepsilon)\cdot\min(w_{\rm a}^{1/\alpha}, T) = (3+\varepsilon)w_{\rm a}^{1/\alpha}$. Since $w_\ell \leq w_{\rm a}$, we also have $w_\ell \leq T^\alpha$.

If $w_\ell > (1-\beta)w_{\rm a}$, then by Lemmas 4.2 and 4.3,

$$\begin{aligned}
\tfrac{{\rm d}\widehat{G}_{\rm a}}{{\rm d}t} + \gamma\tfrac{{\rm d}\Phi}{{\rm d}t} &\leq (w_{\rm a} + s_{\rm a}^\alpha) + \tfrac{\gamma}{\alpha}s_{\rm o}^\alpha + \gamma\cdot(\tfrac{\alpha-1}{\alpha})w_\ell - \gamma\cdot(\tfrac{\alpha}{2\alpha-1})w_\ell^{2-1/\alpha}(3+\varepsilon)w_{\rm a}^{1/\alpha-1} \\
&\leq (1+(3+\varepsilon)^\alpha)w_{\rm a} + \tfrac{\gamma}{\alpha}s_{\rm o}^\alpha + \gamma\cdot(\tfrac{\alpha-1}{\alpha})w_{\rm a} - \gamma\cdot(\tfrac{\alpha}{2\alpha-1})((1-\beta)w_{\rm a})^{2-1/\alpha}(3+\varepsilon)w_{\rm a}^{1/\alpha-1} \\
&\leq \tfrac{\gamma}{\alpha}s_{\rm o}^\alpha + \big((1+(3+\varepsilon)^\alpha) + \gamma\cdot(\tfrac{\alpha-1}{\alpha}) - \gamma\cdot(\tfrac{\alpha}{2\alpha-1})(1-\beta)^{2-1/\alpha}(3+\varepsilon)\big)w_{\rm a}\ .
\end{aligned}$$

Choosing $\gamma = (2-\frac{1}{\alpha})(1+(3+\varepsilon)^\alpha)$, we have $(1+(3+\varepsilon)^\alpha) = \gamma\cdot(\frac{\alpha}{2\alpha-1})$. By the definition of β and the fact that $0<\beta<1$, we have $(1-\beta)^{2-1/\alpha}(3+\varepsilon) > (1-\beta)^2(3+\varepsilon) \geq 3$. Furthermore, $\frac{2\alpha}{2\alpha-1} > 1 > \frac{\alpha-1}{\alpha}$.

Therefore,

$$\begin{aligned}
&1+(3+\varepsilon)^{\alpha}+\gamma(\frac{\alpha-1}{\alpha})-\gamma(\frac{\alpha}{2\alpha-1})(1-\beta)^{2-\frac{1}{\alpha}}(3+\varepsilon)\\
&\leq \gamma(\tfrac{\alpha}{2\alpha-1})(1+2-3)\\
&=0
\end{aligned}$$

Since $\frac{\gamma}{\alpha}=\frac{1}{\alpha}\cdot(2-\frac{1}{\alpha})(1+(3+\varepsilon)^{\alpha})\leq 2(1+(3+\varepsilon)^{\alpha})$ and $c=(\frac{18}{\varepsilon}+4)(1+(3+\varepsilon)^{\alpha})$, it follows that $c\geq\frac{\gamma}{\alpha}$ and thus $\frac{\mathrm{d}\widehat{G}_{\mathrm{a}}}{\mathrm{d}t}+\gamma\frac{\mathrm{d}\Phi}{\mathrm{d}t}\leq\frac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}\leq c\cdot\frac{\mathrm{d}\widehat{G}_{\mathrm{o}}}{\mathrm{d}t}$.

If $w_{\ell}\leq(1-\beta)w_{\mathrm{a}}$, we simply adapt the bound of $\frac{\mathrm{d}\Phi_{\mathrm{a}}}{\mathrm{d}t}$ in Lemma 4.3 as $\frac{\mathrm{d}\Phi_{\mathrm{a}}}{\mathrm{d}t}\leq 0$. Since any active job in WRR that is not lagging must also be an active job in OPT, $w_{\mathrm{o}}\geq w_{\mathrm{a}}-w_{\ell}\geq w_{\mathrm{a}}-(1-\beta)w_{\mathrm{a}}\geq\beta w_{\mathrm{a}}$. Choosing $\gamma=(2-\frac{1}{\alpha})(1+(3+\varepsilon)^{\alpha})$ and recalling that $\beta=\varepsilon/(6+2\varepsilon)$, by Lemma 4.2,

$$\begin{aligned}
\tfrac{\mathrm{d}\widehat{G}_{\mathrm{a}}}{\mathrm{d}t}+\gamma\tfrac{\mathrm{d}\Phi}{\mathrm{d}t} &\leq (1+(3+\varepsilon)^{\alpha})w_{\mathrm{a}}+\tfrac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}+\gamma\cdot(\tfrac{\alpha-1}{\alpha})w_{\ell}\\
&\leq \tfrac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}+\left((1+(3+\varepsilon)^{\alpha})+\gamma\right)w_{\mathrm{a}}\\
&\leq \tfrac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}+\left((1+(3+\varepsilon)^{\alpha})+(2-\tfrac{1}{\alpha})(1+(3+\varepsilon)^{\alpha})\right)w_{\mathrm{a}}\\
&\leq \tfrac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}+3(1+(3+\varepsilon)^{\alpha})\tfrac{w_{\mathrm{o}}}{\beta}\\
&= \tfrac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}+(\tfrac{18}{\varepsilon}+4)(1+(3+\varepsilon)^{\alpha})w_{\mathrm{o}}\\
&\leq c\cdot\tfrac{\mathrm{d}\widehat{G}_{\mathrm{o}}}{\mathrm{d}t}\quad .
\end{aligned}$$

Case 2: $w_{\mathbf{a}}>T^{\alpha}$ **and** $w_{\ell}\leq T^{\alpha}$**.** In this case, $s_{\mathrm{a}}=(3+\varepsilon)\cdot\min(w_{\mathrm{a}}^{1/\alpha},T)=(3+\varepsilon)T$. Since $w_{\ell}\leq T^{\alpha}$ and w_{o} is at least the total weight of non-lagging jobs in WRR (because if a job is non-lagging in WRR, it must be active in OPT), we have $w_{\mathrm{a}}\leq w_{\mathrm{o}}+w_{\ell}\leq w_{\mathrm{o}}+T^{\alpha}$.

If $w_{\ell}>(1-\beta)w_{\mathrm{a}}$, then by Lemmas 4.2 and 4.3,

$$\begin{aligned}
\tfrac{\mathrm{d}\widehat{G}_{\mathrm{a}}}{\mathrm{d}t}+\gamma\tfrac{\mathrm{d}\Phi}{\mathrm{d}t} &\leq (w_{\mathrm{a}}+(3+\varepsilon)^{\alpha}T^{\alpha})+\tfrac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}+\gamma\cdot(\tfrac{\alpha-1}{\alpha})w_{\ell}-\gamma\cdot(\tfrac{\alpha}{2\alpha-1})w_{\ell}^{2-1/\alpha}\tfrac{(3+\varepsilon)T}{w_{\mathrm{a}}}\\
&\leq ((w_{\mathrm{o}}+T^{\alpha})+(3+\varepsilon)^{\alpha}T^{\alpha})+\tfrac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}+\gamma\cdot(\tfrac{\alpha-1}{\alpha})T^{\alpha}-\gamma\cdot(\tfrac{\alpha}{2\alpha-1})((1-\beta)w_{\mathrm{a}})^{2-1/\alpha}\tfrac{(3+\varepsilon)T}{w_{\mathrm{a}}}\\
&\leq w_{\mathrm{o}}+\tfrac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}+\left((1+(3+\varepsilon)^{\alpha})+\gamma(\tfrac{\alpha-1}{\alpha})-\gamma(\tfrac{\alpha}{2\alpha-1})(1-\beta)^{2-1/\alpha}(3+\varepsilon)\right)T^{\alpha}\\
&\quad\text{(since } w_{\mathrm{a}}>T^{\alpha}\text{)}.
\end{aligned}$$

Choosing $\gamma=(2-\frac{1}{\alpha})(1+(3+\varepsilon)^{\alpha})$ and then using the same argument as in Case 1, we can argue that the coefficient of T^{α} is non-positive. Therefore, $\frac{\mathrm{d}\widehat{G}_{\mathrm{a}}}{\mathrm{d}t}+\gamma\frac{\mathrm{d}\Phi}{\mathrm{d}t}\leq w_{\mathrm{o}}+\frac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}\leq c\cdot\frac{\mathrm{d}\widehat{G}_{\mathrm{o}}}{\mathrm{d}t}$.

If $w_{\ell}\leq(1-\beta)w_{\mathrm{a}}$, we simply use the bound $\frac{\mathrm{d}\Phi_{\mathrm{a}}}{\mathrm{d}t}\leq 0$. Note that $w_{\mathrm{a}}>T^{\alpha}$ and $w_{\mathrm{a}}\geq w_{\ell}$. By Lemma 4.2, we have

$$\begin{aligned}
\tfrac{\mathrm{d}\widehat{G}_{\mathrm{a}}}{\mathrm{d}t}+\gamma\tfrac{\mathrm{d}\Phi}{\mathrm{d}t} &\leq (w_{\mathrm{a}}+(3+\varepsilon)^{\alpha}T^{\alpha})+\tfrac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}+\gamma\cdot(\tfrac{\alpha-1}{\alpha})w_{\ell}\\
&\leq \tfrac{\gamma}{\alpha}s_{\mathrm{o}}^{\alpha}+\left((1+(3+\varepsilon)^{\alpha})+\gamma\right)w_{\mathrm{a}}\quad .
\end{aligned}$$

Choosing $\gamma=(2-\frac{1}{\alpha})(1+(3+\varepsilon)^{\alpha})$ and then using the same argument as in Case 1, we can show that $\frac{\mathrm{d}\widehat{G}_{\mathrm{a}}}{\mathrm{d}t}+\gamma\frac{\mathrm{d}\Phi}{\mathrm{d}t}\leq c\cdot\frac{\mathrm{d}\widehat{G}_{\mathrm{o}}}{\mathrm{d}t}$.

Case 3: $w_{\mathbf{a}} > T^\alpha$ and $w_\ell > T^\alpha$. In this case, $s_{\mathrm{a}} = (3+\varepsilon)T \le (3+\varepsilon)w_{\mathrm{a}}^{1/\alpha}$.
If $w_\ell > (1-\beta)w_{\mathrm{a}}$, then by Lemmas 4.2 and 4.3,

$$\begin{aligned}\frac{\mathrm{d}\widehat{G}_{\mathrm{a}}}{\mathrm{d}t} + \gamma\frac{\mathrm{d}\Phi}{\mathrm{d}t} &\le (1+(3+\varepsilon)^\alpha)w_{\mathrm{a}} + \gamma\cdot\left(\tfrac{2\alpha}{2\alpha-1}\right)w_\ell - \gamma\cdot\left(\tfrac{\alpha}{2\alpha-1}\right)(3+\varepsilon)\tfrac{w_\ell^2}{w_{\mathrm{a}}}\\ &\le (1+(3+\varepsilon)^\alpha)w_{\mathrm{a}} + \gamma\cdot\left(\tfrac{2\alpha}{2\alpha-1}\right)w_{\mathrm{a}} - \gamma\cdot\left(\tfrac{\alpha}{2\alpha-1}\right)(3+\varepsilon)\tfrac{((1-\beta)w_{\mathrm{a}})^2}{w_{\mathrm{a}}}\end{aligned}$$

Choosing $\gamma = (2-\frac{1}{\alpha})(1+(3+\varepsilon)^\alpha)$, we have $(1+(3+\varepsilon)^\alpha) = \gamma\cdot(\frac{\alpha}{2\alpha-1})$. By the definition of β, $(1-\beta)^2(3+\varepsilon) \ge 3$. Therefore, $\frac{\mathrm{d}\widehat{G}_{\mathrm{a}}}{\mathrm{d}t} + \gamma\frac{\mathrm{d}\Phi}{\mathrm{d}t} \le \left(\gamma\cdot(\frac{\alpha}{2\alpha-1}) + \gamma\cdot(\frac{2\alpha}{2\alpha-1}) - \gamma\cdot(\frac{3\alpha}{2\alpha-1})\right)w_{\mathrm{a}} = 0 \le c\cdot\frac{\mathrm{d}\widehat{G}_{\mathrm{o}}}{\mathrm{d}t}$.

If $w_\ell \le (1-\beta)w_{\mathrm{a}}$, we simply use the bound $\frac{\mathrm{d}\Phi_{\mathrm{a}}}{\mathrm{d}t} \le 0$. Recall that in this case, $w_{\mathrm{o}} \ge \beta w_{\mathrm{a}}$. Choosing $\gamma = (2-\frac{1}{\alpha})(1+(3+\varepsilon)^\alpha)$ and recalling that $\beta = \varepsilon/(6+2\varepsilon)$, by Lemma 4.2,

$$\begin{aligned}\frac{\mathrm{d}\widehat{G}_{\mathrm{a}}}{\mathrm{d}t} + \gamma\frac{\mathrm{d}\Phi}{\mathrm{d}t} &\le (1+(3+\varepsilon)^\alpha)w_{\mathrm{a}} + \gamma\cdot\left(\tfrac{2\alpha}{2\alpha-1}\right)w_\ell\\ &\le 3(1+(3+\varepsilon)^\alpha)w_{\mathrm{a}}\\ &\le 3(1+(3+\varepsilon)^\alpha)\tfrac{w_{\mathrm{o}}}{\beta}\\ &= (\tfrac{18}{\varepsilon}+4)(1+(3+\varepsilon)^\alpha)w_{\mathrm{o}}\\ &\le c\cdot\frac{\mathrm{d}\widehat{G}_{\mathrm{o}}}{\mathrm{d}t}\quad .\end{aligned}$$

In conclusion, the running condition is satisfied in all the three cases. □

5 Conclusion

In this paper we have given two non-clairvoyant scheduling algorithms for minimizing flow plus energy. The first algorithm (LAPS) is $8\alpha^2$-competitive for (unweighted) flow plus energy, when using a processor with maximum speed $\frac{\alpha}{\alpha-1}T$. The second algorithm (WRR) is $O(3^\alpha/\varepsilon)$-competitive for weighted flow plus energy, when using a processor with maximum speed $(3+\varepsilon)T$, where $0 < \varepsilon \le \frac{3}{\alpha}$. We believe that LAPS can be generalized to minimize weighted flow plus energy, and the competitive ratio would remain $O(\alpha^2)$.

References

[1] S. ALBERS AND H. FUJIWARA: Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms, 3(4):49*, 2007. 1, 2

[2] L. ANDREW, A. WIERMAN, AND A. TANG: Optimal speed scaling under arbitrary power functions. *ACM SIGMETRICS Performance Evaluation Review, 37(2)*, pp. 39–41, 2009. 3

[3] N. BANSAL AND H. L. CHAN: Weighted flow time does not admit $O(1)$-competitive algorithms. In *Proc. SODA*, pp. 1238–1244, 2009. 3

[4] N. BANSAL, H. L. CHAN, T. W. LAM, AND L. K. LEE: Scheduling for speed bounded processors. In *Proc. ICALP*, pp. 409–420, 2008. 1, 2, 3, 4

[5] N. BANSAL, H. L. CHAN, AND K. PRUHS: Speed scaling with an arbitrary power function. In *Proc. SODA*, pp. 693–701, 2009. 1, 2, 3, 4

[6] N. BANSAL, H. L. CHAN, K. PRUHS, AND D. KATZ: Improved bounds for speed scaling in devices obeying the cube-root rule. In *Proc. ICALP*, pp. 144–155, 2009. 2

[7] N. BANSAL, T. KIMBREL, AND K. PRUHS: Speed scaling to manage energy and temperature. *Journal of the ACM, 54(1):3*, 2007. 2

[8] N. BANSAL, K. PRUHS, AND C. STEIN: Speed scaling for weighted flow time. *SIAM Journal on Computing, 39(4)*, pp. 1294–1308, 2009. 1, 2, 4, 5

[9] D. M. BROOKS, P. BOSE, S. E. SCHUSTER, H. JACOBSON, P. N. KUDVA, A. BUYUKTOSUNOGLU, J. D. WELLMAN, V. ZYUBAN, M. GUPTA, AND P. W. COOK: Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro, 20(6)*, pp. 26–44, 2000. 2

[10] H. L. CHAN, W. T. CHAN, T. W. LAM, L. K. LEE, K. S. MAK, AND P. W. H. WONG: Optimizing throughput and energy in online deadline scheduling. *ACM Transactions on Algorithms, 6(1):10*, 2009. 2, 5

[11] H. L. CHAN, J. EDMOND, AND K. PRUHS: Speed scaling of processes with arbitrary speedup curves on a multiprocessor. In *Proc. SPAA*, pp. 1–10, 2009. 1, 3

[12] H. L. CHAN, J. EDMONDS, T. W. LAM, L. K. LEE, A. MARCHETTI-SPACCAMELA, AND K. PRUHS: Nonclairvoyant speed scaling for flow and energy. In *Proc. STACS*, pp. 255–264, 2009. 1, 2, 3, 5

[13] S. H. CHAN, T. W. LAM, AND L. K. LEE: Non-clairvoyant speed scaling for weighted flow time. In *Proc. ESA*, pp. 23–35, 2010. 1

[14] S. H. CHAN, T. W. LAM, L. K. LEE, H. F. TING, AND P. ZHANG: Non-clairvoyant scheduling for weighted flow time and energy on speed bounded processors. In *Proc. CATS*, pp. 3–10, 2010. 1

[15] G. GREINER, T. NONNER, , AND A. SOUZA: The bell is ringing in speed-scaled multiprocessor scheduling. In *Proc. SPAA*, pp. 11–18, 2009. 1

[16] S. IRANI, S. SHUKLA, AND R. K. GUPTA: Algorithms for power savings. *ACM Transactions on Algorithms, 3(4):41*, 2007.

[17] B. KALYANASUNDARAM AND K. PRUHS: Minimizing flow time nonclairvoyantly. *Journal of the ACM, 50(4)*, pp. 551–567, 2003. 3

[18] T. W. LAM, = L. K. LEE, I. K. K. TO, AND P. W. H. WONG: Speed scaling functions for flow time scheduling based on active job count. In *Proc. ESA*, pp. 647–659, 2008. 1, 2, 3, 4

[19] T. W. LAM, L. K. LEE, H. F. TING, I. K. K. TO, AND P. W. H. WONG: Sleep with guilt and work faster to minimize flow plus energy. In *Proc. ICALP*, pp. 665–676, 2009. 1, 2

[20] T. W. LAM, L. K. LEE, I. K. K. TO, AND P. W. H. WONG: Competitive non-migratory scheduling for flow time and energy. In *Proc. SPAA*, pp. 256–264, 2008. 1, 2

[21] R. MOTWANI, S. PHILLIPS, AND E. TORNG: Nonclairvoyant scheduling. *Theoretical Computer Science, 130(1)*, pp. 17–47, 1994. 3

[22] T. MUDGE: Power: A first-class architectural design constraint. *IEEE Computer, 34(4)*, pp. 52–58, 2001. 2

[23] J. M. STEELE: *The Cauchy-Schwarz master class: An introduction to the art of mathematical inequalities*, p. 136. Cambridge University Press, 2004. 7

[24] F. YAO, A. DEMERS, , AND S. SHENKER: A scheduling model for reduced cpu energy. In *Proc. FOCS*, pp. 374–382, 1995. 2

AUTHORS

Sze-Hang Chan
Department of Computer Science
University of Hong Kong, Hong Kong

Tak-Wah Lam
Department of Computer Science
University of Hong Kong, Hong Kong

Lap-Kei Lee
Max-Planck-Institut für Informatik
66123 Saarbrücken, Germany

Hing-Fung Ting
Department of Computer Science
University of Hong Kong, Hong Kong

Peng Zhang
School of Computer Science and Technology
Shandong University, China